Genetic Algorithms and Image Search

Pavel Mrázek

Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University (ČVUT),
Karlovo nám. 13, 12135 Praha 2, Czech Republic
e-mail: xmrazek@sun.felk.cvut.cz

Abstract: This article is intended to provide an introduction to Genetic Algorithms (GAs), concentrating on their abilities of function optimization. Basic features of GAs are presented in the first part. The second part describes one application: GAs are employed to search images for instances of known objects.

Keywords: genetic algorithms, function optimization, model-based image search

1. Introduction

If we are to optimize a function, we have various methods to choose from to accomplish the task. The traditional techniques include calculus-based, enumerative, and random search algorithms.

Calculus-based methods make often use of function derivatives and thus are restricted to smooth functions. Although highly efficient on a specific problem domain, they cannot be used generally, because in many real-world problems the function to be optimized is noisy and discontinuous, and so unsuitable for calculus-based algorithms.

Enumerative and random walk techniques have one common advantage, but also one common drawback: they are simple, but desperately inefficient, and therefore inappropriate for larger search space.

We would like to have an optimization method suitable for a wide problem domain, for complex and large scale space, insensitive to noise and discontinuities. We are willing to sacrifice the peak single-problem performance of a calculus-based procedure, but demand generality and robustness. Genetic Algorithms are suggested as a method which meets these requirements (e.g. [1], [2]).

2. What are Genetic Algorithms

2.1 Basic Features

As D. Goldberg states in his book [1], GAs are different from traditional techniques in four ways:

- "GAs work with a coding of the parameter set, not the parameters themselves." Parameters are encoded as bit strings. This encoding makes GAs independent of a particular problem and allows to employ principally the same technique to solve problems of different complexity. The bit strings are often called *chromosomes*, as analogy with the information-carrying objects in living cells. GAs draw much inspiration from the world of natural genetics.
- "GAs search from a population of points, not a single point." A population of N chromosomes is maintained. Since each chromosome represents one solution, we process N possible solutions at a time. This makes GAs more resistant to false peaks (local extreme) of the objective function.

- "GAs use objective function information, not derivatives or other auxiliary knowledge." Relying only on the function value, the method gains generality. Unlike the above mentioned calculus-based techniques, GAs do not require the existence of a gradient, and are therefore not limited to continuous or smooth functions.
- "GAs use probabilistic transition rules, not deterministic rules." Last but not least, this difference is inherent in the mechanism of creating a new generation. As mentioned before, GAs maintain a population of N chromosomes. The solution is obtained iteratively, imitating the behavior of a natural system, and creating new generations from the precedent using genetic operators *crossover* and *mutation*. Crossover takes two parent chromosomes (chosen from the parent population), chooses a random crossing site and swaps parts of the chromosomes (see Figure 1). Mutation switches a bit of a chromosome. Which two chromosomes should mate and why these operators lead to a solution is explained below.

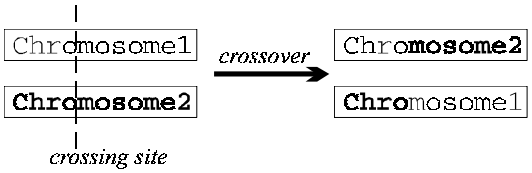

Figure 1*: Pair of chromosomes before and after crossover. Crossing site is chosen at random and parts of the chromosomes are exchanged.*

If we said before that probabilistic transition rules are used and crossing site for crossover is chosen at random, we must distinguish between the random techniques defamed in the introduction, and the randomized operators of GAs. The latter work very well and yield good results thanks to strategy called figuratively *selective breeding*. This means that the higher value of the objective function an individual had in the parent generation, the more descendants it will have in the next generation. Below average members will be likely to have less then average number of children, and so bad solutions will gradually die off. Then, crossover assures an exchange of information between the individuals, takes parts of the parents to combine them and create possibly fitter children. Mutation serves as an insurance against loss of information. Analogy of this process to natural selection is clear: the fittest members of a population mate to bore strong children.

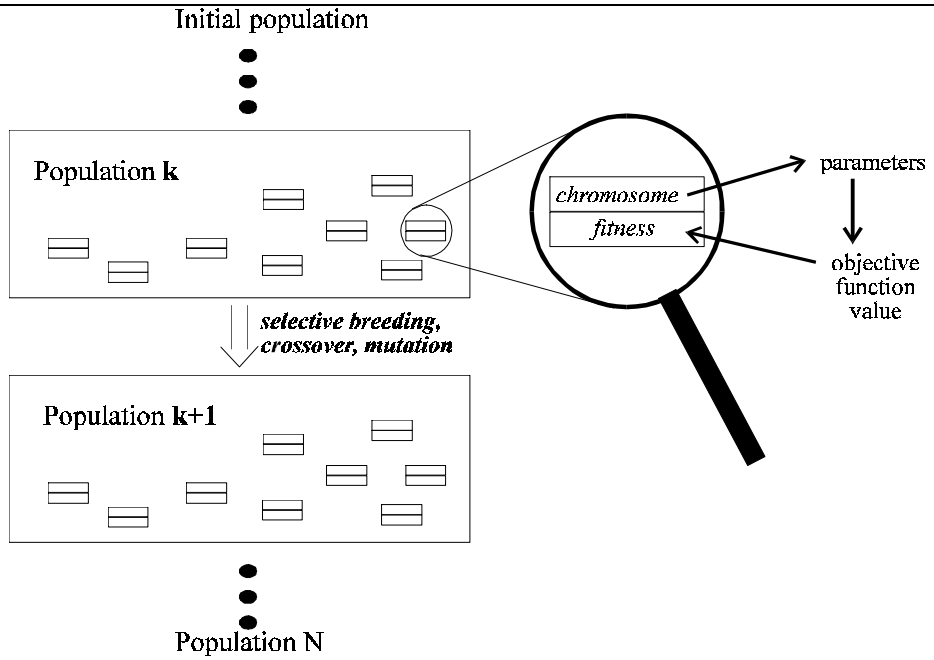

Figure 2*: Draft of the Genetic Algorithms - populations evolve from the initial through selective breeding and genetic operators, parameter decoding and objective function evaluation link the mechanism to a specific task.*

Chromosome No.	Initial population	x value	f(x) = x^3	p_selection f/sum(f)	Actually selected
1	0 0 1 0 1	5	125	0,00	0
2	1 0 0 0 1	17	4913	0,17	1
3	1 1 0 0 0	24	13824	0,47	2
4	1 0 1 1 0	22	10648	0,36	1
Sum			29510	1,00	4
Average			7378	0,25	1
Max			**13824**	0,47	2

Table 1: GAs at work. Initial population, objective function values and selection of chromosomes to form the next generation.

Selected chromosomes	Mate (randomly	Crossover site selected)	New population	x value	f(x) = x^3
1 0 0 0 1	2	4	1 0 0 0 0	16	4096
1 1 0 0 0	1	4	1 1 0 0 1	25	15625
1 1 0 0 0	4	3	1 1 0 1 0	26	17576
1 0 1 1 0	3	3	1 0 1 0 0	20	8000
Sum					45297
Average					11324
Max					**17576**

Table 2: GAs at work. Chromosomes selected for mating, crossover and the new population. (Note: probability of crossover was set to 1.0, probability of mutation was 0.05 and didn't occur.)

2.2 GAs at work

Let's try a very simple simulation to clarify how GAs work. Imagine we have to optimize the function $f(x) = x^3$, for x being from the interval <0, 31>. First, we must choose the encoding of the parameter. Here the solution is straightforward, x will be stored as five-bit unsigned integer.

The optimization process starts with the initial population, randomly generated. At the beginning we simply fill the chromosomes (bit strings) with 0's and 1's at random. Then, to link the population to a specific problem, objective function values are computed for the chromosomes. Note that chromosome decoding and objective function are the only elements specific for a problem. The rest of the mechanism stays generally the same for every task (see Figure 2).

When each member of the population has its fitness (objective function) value assigned, we can move on to create the next generation. To do this, we have first to decide which individuals are going to take part in the process of creation. We employ the strategy of selective breeding in this way: probability of selection is computed for every chromosome based on its relative fitness value. Then weighted coin toss is performed for each chromosome in turn. If the coin shows 'yes' (it does it with the corresponding probability of selection), the chromosome is reproduced into a group of prospective parents. The selection procedure is repeated until the size of the group is equal to the size of the generation. Best individuals are likely to have more than one copy in the group. Table 1 gives an example: initial population, fitness values, probability of selection and actual number of selected chromosomes.

Having selected the parents for the following generation, crossover is the next step to come. Individuals mate randomly and crossover swaps parts of their bit strings to create a new generation. The result with the corresponding decoded parameters and the objective function values is shown in Table 2.

If you compare the populations in Table 1 and Table 2, an important observation emerges: only by applying the selective breeding strategy and by swapping parts of the chromosomes, we

were able to improve the maximum, as well as the average objective function values. This is the principle of Genetic Algorithms. Repeat this selection and mating N times for a population of a reasonable size, and you have a fair chance of obtaining the optimal result.

2.3 Why do GAs work so well?

How is it possible, that we should get something valuable only by combining parts of strings? Well, if we imagine that substrings of the chromosomes can represent solutions to subproblems of our task, it becomes quite natural to match these substrings to find a global solution. In [1] again, a quotation from mathematician J. Hadamard is found:
"... it is obvious that invention or discovery, be it in mathematics or anywhere else, takes place by combining ideas."
But it is not only this feeling that somewhat humanlike behavior should lead to success, subtle mathematics lies in the foundations of Genetic Algorithms as well. Probably the most important notion is that of schemata and building blocks. Detailed explanation is beyond the scope of this article and can be found in Goldberg's book [1]. The principal idea is that number of highly fit, short substrings (building blocks) will increase exponentially, because better individuals get more trials and short blocks are not likely to be cut by crossover (because the crossing site is chosen at random, short building blocks have better chance to be missed and survive than longer strings).

3. Genetic Algorithms Applied to Image Search

When searching for an instance of known object in an image, a model of the object may be used profitably. Then we have to find such a position of the model in an image, where the object appearance stored for the model best matches the image data. If this model-to-data fitness is expressed in terms of an objective function, the task is transformed into a problem of optimizing the function. This fitness function is likely to be noisy and discontinuous, and therefore suitable for GAs.
Various models are found in literature, some of them, applicable to the field of magnetic resonance image segmentation (the problem we are tackling), have been summarized in [3]. We tried to implement a model similar to the one presented in [4].
To mention the basic features of the model:
The shape of an object is modelled as a set of connected points, forming the model boundary. The grey-level appearance of the model is stored for the boundary points. Both shape and appearance are trained on examples of the object. The model is parametrical, and is allowed to change position, scale, rotation and shape, being limited by constraints learnt during the training.
The objective (fitness) function of the model projected into the image data is computed as a sum of fitnesses of individual model points. For each point, we look in its neighbourhood to find the location, where the appearance stored for the point best corresponds to the image. Then the fitness is directly proportional to quality of the fit (weight), and inversely proportional to the distance between the current and the best fit position (displacement). The algebraic form follows:

$$obj_function = \sum_{for_all_p.} p_fitnesses,$$

$$\text{where} \qquad p_fitness = weight * \left(const1 + \frac{const2}{1 + displacement} \right).$$

We chose const1=30 and const2=70, to have the maximum possible fitness equal to 100. The member const1 was placed into the expression after empirical experiments, so that the

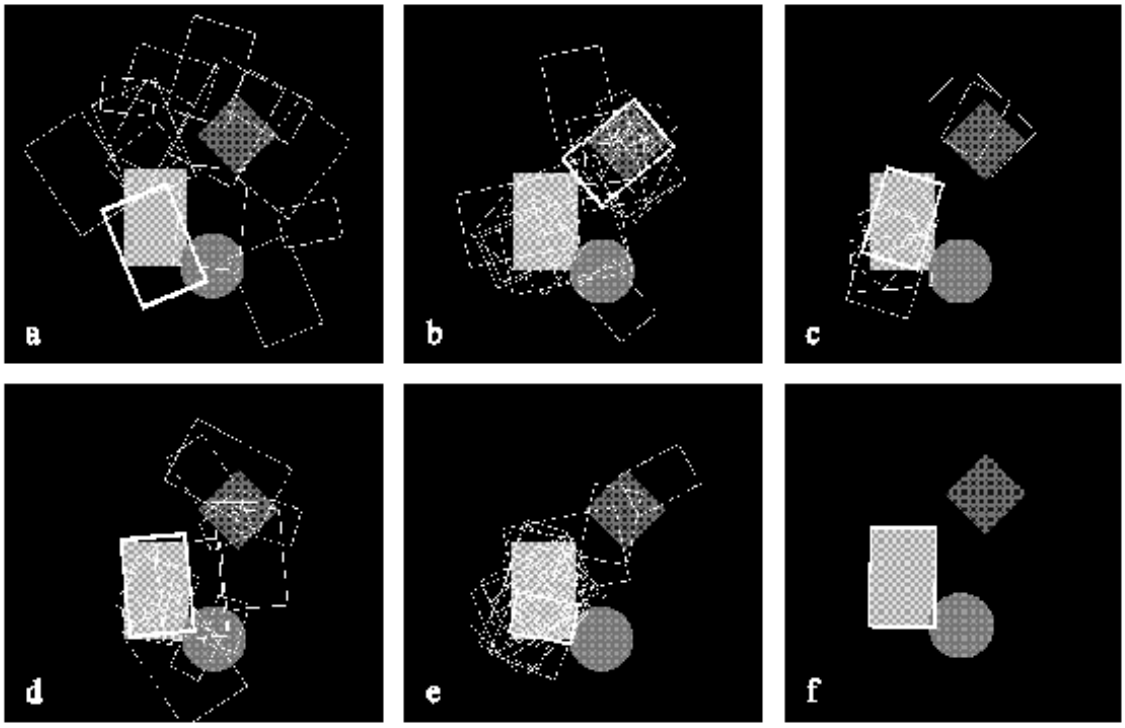

Figure 3: Genetic Algorithms used for model-based image search. The objects in the scene are shown in grey, instances of the model in white. Bold white line shows the instance with highest fitness value of the generation. Pictures show various stages of the solution: a) several members of the initial generation; b), c), d), e) above average individuals of generations 10, 20, 35 and 50, respectively; f) final solution.

contribution of points, where good fit was encountered, is not so dramatically decreased by the displacement member.

Up to now, functionality of the algorithm has been tested on artificial data (such as the scene in Figure 3), with model of four parameters - x and y position, scale and rotation.

In the experiment being described here, the model was trained to represent a rectangle with side ratio 2:3, lighter than background. The scene was 512x512 pixels, the x and y coordinates of the model center were limited to the interval <100, 400>. The scale of the model could vary between 0.6 and 1.5 of the original size, no constraints were put on the rotation parameter. We used 25 bits to encode these model parameters. Concerning the GAs' settings, the probability of crossover was 0.6, and the probability of mutation 0.03. Size of the populations varied between 40 and 150 individuals (see graphs below), number of generations was limited to 50.

A typical run can be seen in Figure 3. At the beginning (a), individuals are spread randomly in the search space. As new generations are created, chromosomes are combined and combinations of parameters investigated (b to e). Gradually, more individuals concentrate in the area of main interest (i.e. of higher values of objective function. The question of choosing a good objective function is crucial.)

Note that the globally best solution, even if discovered in a generation, is not guaranteed to propagate through the generations to the final one. It may be easily destroyed by crossover or mutation. Compare pictures d) and e) in Figure 3 - a very good estimate of the best solution appeared in the 35th generation, but didn't survive till generation 50. This has one advantage:

the algorithms resists false peaks of local maxima, and keeps trying to locate a better solution. The remedy is simple, we have to store the best result over generations.

Another remark: we declared above that GAs use only the objective function value, no other information (gradient etc.), and are therefore not limited to smooth functions. However, if we are able to exploit more information from the function, it is advisable to do so. This is the case; if a good match between the model and the data is encountered, an even better match is likely to be found by finely tuning the model parameters. A description of how to improve the model-to-data fit locally, and how to link this technique to the mechanisms of GAs has been published (see [5] and [6], respectively). Similar hybrid schemes were mentioned in [1], and we can quote: "... genetic algorithm finds a hill and the hill-climber (a local technique) goes and climbs it." Picture f) of Figure 3 shows the result of such an approach: approximate solution found by GAs was refined by a local technique. Our model located the rectangle in the image data, even though the rectangle was partially occluded by a circle.

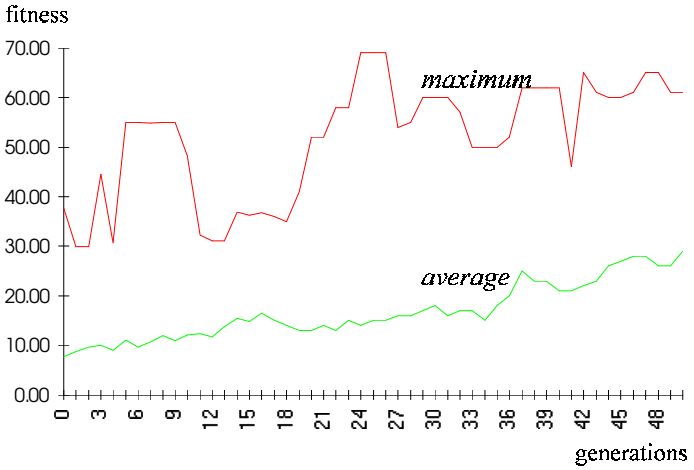

Figure 4: Development of maximum and average fitness values for the image search problem. Each generation in this experiment consisted of 50 individuals.

Figure 4 shows an example of how maximum and average fitness values behave during the generations evolution for our image search problem. The maximum fitness jumps up and down as highly fit members are created and destroyed. The average fitness value of a population grows steadily, and pushes also the maximum value up.

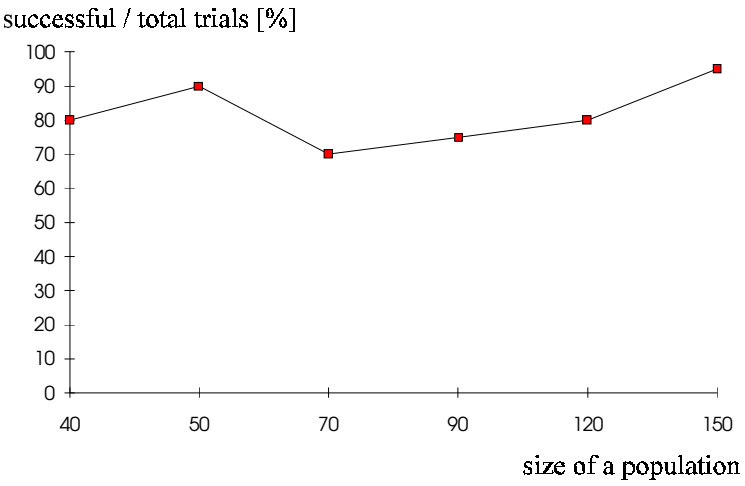

Figure 5: Success ratio for populations of different sizes. Number of generations was kept to 50.

We have to say that Genetic Algorithms are not guaranteed to find the optimal result. We conducted an experiment, varying the size of the populations for our image search problem between 40 and 150 individuals. The number of iterations was limited to 50. The success ratio was from 75 to 90 percent. The bigger populations, the higher the probability of success should be; however, it didn't appear so. The larger populations would probably need more generations to try more combinations of the members.

4. Conclusion

In this text we tried to provide an introduction to Genetic Algorithms, a powerful method of function optimization. We described the basic features, the mechanism and elementary operators - crossover and mutation. A very simple example was presented to show how new generations are created.

In the second part, application of GAs to the domain of image search was introduced. We mentioned that this specific task is linked to the genetic mechanism through parameter encoding and objective function evaluation. Results of our implementation were presented and discussed.

5. Acknowledgments

The author would like to thank his colleagues Ruda Lecjaks for his help with implementing the graphical output in the GL library and Petr Felkel for technical and organizational support.

6. References

[1] D. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley 1989

[2] A. Hill, C. J. Taylor: Model-Based Image Interpretation Using Genetic Algorithms, Proc. British Machine Vision Conference, Springer-Verlag 1991

[3] P. Mrázek, P. Felkel, L. Sýkora: Model-Based Segmentation of Medical Images, 11. Spring Conference on Computer Graphics, SCCG'95, Bratislava, Proceedings

[4] Cootes, Hill, Taylor, Haslam: The Use of Active Shape Models For Locating Structures in Medical Images. Barrett, Gmitro (Eds.): Information Processing in Medical Imaging. Springer-Verlag, Lecture Notes in Computer Science 1993

[5] T. F. Cootes, C. J. Taylor: Active Shape Models - 'Smart Snakes', Proc. British Machine Vision Conference, Springer-Verlag 1992

[6] A. Hill, T. F. Cootes, C. J. Taylor: A Generic System for Image Interpretation Using Flexible Templates, Proc. British Machine Vision Conference, Springer-Verlag 1992

This research has been conducted as a part of "VirtMed" project and has been supported by ČVUT grant No. 10038280.